IDENTIFIKASI BENDA MENGGUNAKAN KAMERA BERBASIS ANDROID

Dedy Gunawan[1)], AntoniusWibowo[2)]
Email: desta_franz007@yahoo.com

ABSTRAK

Dewasa ini, Android adalah salah satu sistem operasi yang sangat populer di masyarakat. Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar. Android menyediakan platform terbuka bagi para pengembang, sehingga pengguna bisa membuat aplikasi baru di dalamnya. Aplikasi yang mereka ciptakan sendiri bisa untuk digunakan pada bermacam peranti bergerak.

Pada penelitian ini, perangkat yang akan diintegrasikan menggunakan Android adalah sebuah robot humanoid berbasis mikrokontroler CM-510. Robot humanoid ini bergerak berdasarkan sinyal IR yang diterimanya dari sebuah remote control yang mengirimkan sinyal IR tersebut. Perancangan yang diperlukan adalah perancangan hardware meliputi pembuatan RS-232 yang berfungsi sebagai komunikasi robot humanoid dengan handphone Android, lalu pembuatan mekanika gerak kepala robot agar kepala robot dapat bergerak. Untuk perancangan software meliputi pembuatan aplikasi handphone Android, dan pemrograman humanoid.

Dengan adanya teknologi ini, diharapkan robot humanoid tersebut dapat bergerak secara otomatis mencari, mendekati, dan menendang sebuah benda yang menjadi objek tujuannya. Modul interface yang digunakan untuk menghubungkan Android dengan robot humanoid tersebut adalah Modul IOIO. Modul IOIO ini memiliki sebuah port USB, dan beberapa port I/O. Port USB akan menerima input dari Android dan memberikan output melalui I/O port yang nantinya akan dihubungkan ke robot humanoid tersebut.

Kata kunci: Identifikasi, benda, kamera, Android, *IOIO*, mikrokontroler, robot *humanoid*

PENDAHULUAN

Dewasa ini, Android adalah salah satu sistem operasi yang sangat populer di masyarakat. Android merupakan sistem operasi yang berbasis linux untuk telepon seluler seperti: telepon pintar, dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang, sehingga pengguna bisa membuat aplikasi baru di dalamnya. Aplikasi yang diciptakan sendiri bisa untuk digunakan pada bermacam peranti bergerak. Oeh karena keunggulannya yang menyediakan *platform* terbuka, maka penulis membuat sebuah aplikasi tertentu menggunakan kamera dari Android tersebut. Aplikasi yang akan dibuat menggunakan bahasa pemrograman *Java*.

Di lain sisi, teknologi robot selama ini dapat membantu manusia di berbagai bidang terutama bidang industri. Untuk mengembangkan dunia robotika lebih jauh, para peneliti mencoba membuat robot untuk melakukan berbagai macam olahraga dan permainan yang dibuat manusia. Harapannya, agar robot dapat bertindak seperti manusia dan suatu saat dapat digunakan dalam lingkup yang lebih luas.

Pada penelitian ini, penulis mengintegrasikan kamera dari *handphone* Android dengan sebuah robot *humanoid* yang menggunakan mikrokontroler untuk menggerakkan motor *servo*-nya. *Interface* antara kamera Android, dan robot *humanoid* menggunakan *IOIO for* Android. *IOIO for* Android ini adalah sebuah rangkaian elektronik yang menjadi penghubung Android dengan peranti lain dari luar. Robot *humanoid* awalnya bergerak berdasarkan sinyal *IR* yang diterimanya dari sebuah *remote control* yang yang mengirimkan sinyal *IR* tersebut.

Kamera dari Android dapat berfungsi sebagai mata dari robot *humanoid* tersebut, dan membuat robot yang awalnya bergerak dengan menggunakan *remote control* menjadi bergerak secara otomatis, harapannya adalah robot dapat bertindak lebih menyerupai manusia. Masalah yang muncul dalam pengerjaan alat adalah:

1. Membuat sebuah aplikasi baru untuk Android;
2. Peletakan kamera berbasis Android pada robot *humanoid* dengan tetap menjaga keseimbangan dari robot *humanoid* tersebut;
3. Memahami kode-kode yang dikirimkan *remote control* untuk menggerakkan robot *humanoid*;
4. Memprogram gerakan-gerakan baru pada robot *humanoid* tersebut.

Agar sistem ini lebih spesifik dan terarah, maka pembahasan masalah dalam program ini memiliki batasan-batasan sebagai berikut:

1. Benda yang menjadi target adalah bola bulat berwarna *orange*;

[1)]Mahasiswa di Fakultas Teknik Jurusan Teknik Elektro Universitas Katolik Widya Mandala Surabaya
[2)] Staf Pengajar di Fakultas Teknik Jurusan Teknik Elektro Universitas Katolik Widya Mandala Surabaya

2. Spesifikasi dari *Embedded system* yg akan digunakan (kamera 5 *MP*, 512 *MB RAM*, *CPU* 1 GHz, *OS* Android 2.3 *Gingerbread*);
3. Pengujian dilakukan di sebuah lapangan yang terbuat dari karpet berwarna hijau berukuran 400 cm x 300 cm;
4. Posisi awal robot berada di tengah-tengah lapangan, kemudian akan mencari, mendatangi, dan menendang objek targetnya dengan cara menggerakkan *servo-servo* yang terpasang pada robot *humanoid* tersebut;
5. Pada lapangan akan diletakkan bola-bola berukuran sama namun warna berbeda dari target (putih, merah, biru, hitam).

Tujuan yang hendak dicapai dalam penelitian ini ialah robot *humanoid* dapat bergerak secara otomatis mencari, dan menemukan serta menendang sebuah objek yang menjadi targetnya dengan kamera berbasis Android sebagai mata dari robot.

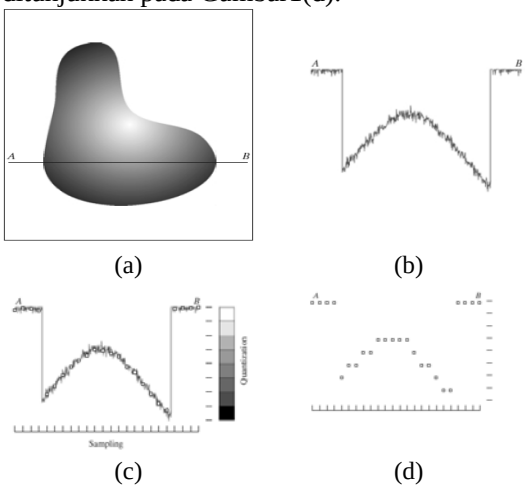
TINJAUAN PUSTAKA

Digital Image Processing

Dalam teori pemrosesan citra terdapat banyak sekali topik yang dipelajari. Pada konteks penelitian ini hanya beberapa topik yang dibahas/digunakan sebagai pedoman seperti: *image sampling and quantization*, segmentasi warna, dan *thresholding.*

Untuk mengkonversi objek yang diindera oleh sensor menjadi citra digital diperlukan dua proses yaitu: ***s****ampling*, dan *quantization.* Teori dasar *sampling* dan kuantisasi diilustrasikan pada Gambar 1. Gambar 1 (a) menunjukkan gambar kontinyu, f(x,y), yang akan di*konversi* kebentuk digital. Untuk mengubahnya menjadi bentuk digital, harus mengambil sampel dari fungsi pada koordinat maupun amplitudo. Digitalisasi nilai koordinat disebut *sampling*. Kuantisasi adalah suatu proses mendigitasi intensitas sinyal objek pada koordinat *pixel* yang di-*sampling*. Dengan kata lain, memberi nilai *pixel* tersebut.

Fungsi satu dimensi ditunjukkan pada Gambar 1 (b) adalah *plot* nilai amplitudo (*gray-level*) dari gambar kontinyu di sepanjang garis AB pada Gambar 1(a). Untuk sampel fungsi ini, diambil sampel seperti *space* di sepanjang garis AB, seperti ditunjukkan pada Gambar 1(c). Lokasi masing-masing sampel diberikan dengan tanda centang vertical di bagian bawah gambar. Sampel ditampilkan sebagai kotak putih kecil ditumpangkan pada fungsi. Himpunan lokasi diskrit tersebut memberikan fungsi sampel. Namun demikian, nilai dari sampel masih mencakup (secara vertikal) berbagai kesinambungan nilai *gray-level.* Untuk membentuk fungsi digital, nilai-nilai *gray-level* juga harus di*konversi*(terkuantisasi) ke dalam jumlah diskrit. Sisi kanan Gambar 1(c) menunjukkan skala nilai *gray-level* terbagi menjadi delapan tingkat diskrit, mulai dari hitam sampai putih. Tanda centang vertical menunjukkan nilai tertentu yang ditugaskan kepada masing-masing nilai *gray-level*. Nilai *gray-level* terus menerus dikuantisasi hanya dengan menetapkan salah satu dari delapan nilai *gray-level* diskrit untuk setiap sampel. Hal tersebut dilakukan tergantung pada jarak vertikal dari sampel ke tanda centang vertikal. Sampel digital yang dihasilkan dari kedua *sampling* dan kuantisasi ditunjukkan pada Gambar 1(d). Kedua *sampling* dan kuantisasi ditunjukkan pada Gambar1(d).



(a) (b)

(c) (d)

Gambar 1. (a) Gambar kontinyu;
(b) Hasil *scan* garis A-B pada Gambar Kontinyu;
(c) *Sampling* dan kuantisasi.
(d) *Scan* garis digital

Segmentasi adalah sebuah proses pembagian sebuah citra menjadi daerah-daerah berdasarkan sifat-sifat tertentu. Pada umumnya semua warna dapat diperoleh dengan mengatur proporsi dari tiga komponen warna saja. Ketiga komponen warna ini terdiri dari: merah (*R*), hijau (*G*), dan biru (*B*) (disebut dengan warna primer). Salah satu kelemahan dari sistem warna *RGB* ini yaitu mudah dipengaruhi oleh cahaya. Sistem warna *HSV* merupakan warna pelengkap yang dihasilkan dari pengaturan warna *RGB. Hue* menyatakan *spectrum* warna dominan cahaya dan menyatakan kejernihan *spectrum* warna dalam cahaya. Semakin jernih suatu warna, maka nilai *saturasi*-nya semakin tinggi. *Value* atau *brightness* menyatakan tingkat kecerahan cahaya dari warna yang

dilihat dari batas yang paling redup atau gelap sampai batas yang paling terang.

Thresholding adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih, sehingga dapat diketahui daerah mana yang termasuk objek dan *background* dari citra secara jelas. Citra hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan objek serta ekstraksi fitur. Metode *thresholding* secara umum dibagi menjadi dua, yaitu:

1. *Thresholding* global
   Thresholding dilakukan dengan mempartisi histogram dengan menggunakan sebuah *threshold* (batas ambang) global T, yang berlaku untuk seluruh bagian pada citra.
2. *Thresholding* adaptif
   Thesholding adaptif dilakukan dengan membagi citra menggunakan beberapa sub citra. Lalu pada setiap sub citra, segmentasi dilakukan dengan menggunakan *threshold* yang berbeda[1].

Handphone OS Android

Android adalah sistem operasi yang berbasis linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang, sehingga pengguna bisa membuat aplikasi baru di dalamnya. Aplikasi-aplikasi yang dibuat menggunakan bahasa pemrograman *Java*. Aplikasi yang dibuat dapat digunakan untuk menggerakkan peranti lainnya dari luar[2].

Aplikasi yang dibuat menggunakan bahasa pemrograman *Java* terlebih dahulu diekspor ke dalam *format* ".APK" agar dapat diinstal pada *handphone OS* Android. Setelah aplikasi diinstal, maka program baru dapat dioperasikan. Aplikasi yang dibuat dalam pengerjaan penelitian ini memanfaatkan kamera dari *handphone* tersebut untuk melakukan *capturing video* dan kemudian dilakukan *image processing.* Setelah itu *handphone* tersebut akan mengirimkan *instruction packet* tertentu dalam bentuk *RS*232 pada *IOIO*.

IOIO

IOIO memiliki *MCU* tunggal yang berfungsi sebagai *host USB* dan menterjemahkan perintah dari sebuah aplikasi Android. Selain itu, *IOIO* dapat berinteraksi dengan perangkat periferal lainnya dengan cara yang sama karena mempunyai *MCU*. Digital *Input/Output*, *PWM*, *Input* Analog, *I2C*, *SPI*, dan semua kontrol *UART* dapat digunakan dengan *IOIO* tersebut. Kode untuk mengontrol antarmuka ini ditulis dalam cara yang sama seperti menulis sebuah aplikasi Android. Dengan kata lain, dapat menggabungkan kekuatan komputasi mengagumkan, konektivitas *bluetooth*, layar sentuh, dan berbagai sensor dari perangkat Android dengan kemampuan untuk mempermudah menambahkan perangkat periferal untuk berinteraksi dengan dunia luar. Dengan menggunakan *IOIO* tidak memerlukan perangkat keras atau memodifikasi perangkat lunak untuk perangkat lainnya[3].

RS-232

Komunikasi serial terbagi menjadi 2, yaitu:

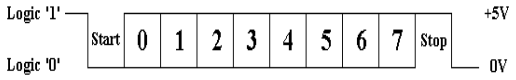
- Komunikasi serial sinkron, yaitu data dikirim bersamaan dengan sinyal *clock*. Contoh protokol yang menggunakan metode ini adalah *Inter-Integrated Circuit* (*I2C*), dan *Serial Peripheral Interface* (*SPI*).
- Komunikasi serial asinkron, yaitu data dikirim tanpa sinyal *clock* karena data dikirimkan dengan kecepatan tertentu yang sama pada pengirim, dan penerima. Contoh protokol yang menggunakan metode ini adalah *Universal Asynchronous Receiver-Transmitter* (*UART*).

Pada *UART*, kecepatan pengiriman data (atau yang sering disebut dengan *Baud Rate*), dan fase *clock* pada sisi *transmitter*, dan sisi *receiver* harus sinkron. Untuk itu diperlukan sinkronisasi antara *transmitter* dan *receiver*. Hal ini dilakukan oleh bit "*Start*" dan bit "*Stop*". Ketika saluran transmisi dalam keadaan *idle*, *output UART* adalah dalam keadaan logika "1".

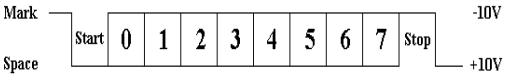
Ketika *transmitter* ingin mengirimkan data, *output UART* akan diatur dulu ke logika "0" untuk waktu satu bit. Sinyal ini pada *receiver* akan dikenali sebagai sinyal "*Start*" yang digunakan untuk menyinkronkan fase *clock*-nya, sehingga sinkron dengan fase *clock transmitter*. Selanjutnya data akan dikirimkan secara serial dari bit yang paling rendah (bit 0) sampai bit tertinggi. Selanjutnya akan dikirimkan sinyal "*Stop*" sebagai akhir dari pengiriman data serial.

Perbedaan *RS*-232 dengan komunikasi serial *TTL*/*CMOS* adalah pada level tegangannya. *IC Max*-232 adalah *IC* yang berfungsi merubah sinyal *RS*-232 menjadi *TTL*, yang dapat dibaca oleh mikrokontroler. *Max* 232 memiliki 2 jalur penerima data (*RXd*) dan 2 jalur pengirim data (*TX*). *IC Max*-232

beroperasi dengan baik menggunakan tegangan 5 V dan arus 8 mA. Gambar 2 dan 3 merupakan perbandingan level tegangan pada *RS*-232 dengan *TTL*/*CMOS*[4].



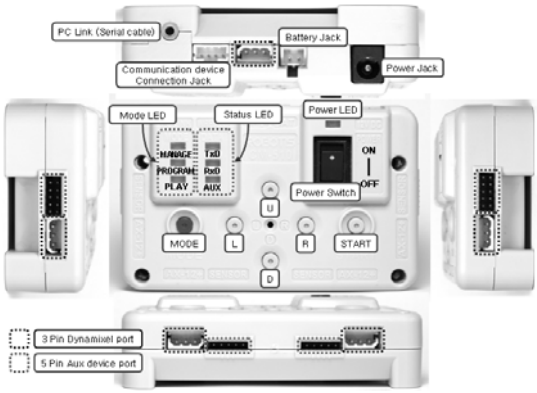
Gambar 2. Level tegangan pada *TTL/CMOS*



Gambar 3. Level tegangan pada *RS*-232

CM-510

CM-510 adalah sebuah mikrokontroler yang diperlengkapi koneksi untuk *dynamixel AX series*, sensor jarak, sensor sentuh, sensor *IR*, dan *device* lainnya. Mikrokontroler ini bekerja pada tegangan 6,5-15 Volt, dianjurkan pada tegangan 11,1 Volt (*Li-PO* 3 *cell*). Bekerja pada suhu -5~70^{o}C. Untuk menghubungkan *CM*-510 dengan *PC* digunakan *jack* 2,5 mm *to* serial. Koneksi tersebut dapat digunakan untuk men-*download* program dari *PC* ke *CM*-510. Gambar 4 merupakan gambar dari *CM*-510.



Gambar 4. *CM*-510

Penjelasan bagian-bagian dari *CM*-510 pada Gambar 4:

- PC *Link (Serial Cable)*: digunakan untuk menghubungkan *CM*-510 dan *PC* melalui *port* serial atau untuk men-*download* program;
- *Communication Device Connection Jack*: digunakan untuk komunikasi *wireless* dari *ZIG*-110 atau menerima sinyal dari *remote control*;
- *Battery Jack*: koneksi ke baterai;
- *Power Jack*: digunakan untuk menghubungkan *SMPS power supply*;
- *Power LED*: menunjukkan *status ON/OFF* robot;
- *Power Switch*: tombol *ON/OFF*;
- *MODE Button*: digunakan untuk menentukan *mode* yang diinginkan;
- *START Button*: digunakan untuk melakukan *mode* yang telah dipilih;
- *U/L/D/R Button*: digunakan untuk melakukan suatu kode perintah tertentu;
- *AX*/*MX Series Bus Port*: digunakan untuk koneksi *AX*/*MX dynamixel* secara seri;
- *Peripheral Devices Connection Port*: digunakan untuk sensor jarak, sensor sentuh, sensor *IR*, dan perlengkapan periferal lain;
- *Mode Display LED*: indikator yang menunjukkan kondisi mode dari *CM*-510[5].

Dynamixel AX-12

Dynamixel seri ini adalah aktuator yang cerdas, modular ini menggabungkan *reducer gear*, motor *DC* yang presisi dan sebuah sirkuit kontrol dengan fungsionalitas jaringan, semua dalam satu paket. Gambar 5 merupakan *Dynamixel* seri.

Gambar 5. *Dynamixel AX*-12

Meskipun ukurannya kecil, tetapi dapat menghasilkan torsi tinggi dan dibuat dengan bahan berkualitas tinggi untuk memberikan kekuatan yang diperlukan dan ketahanan struktural untuk menahan kekuatan eksternal yang besar. *Servo* ini juga memiliki kemampuan untuk mendeteksi dan bertindak berdasarkan kondis iinternal seperti perubahan suhu internal atau tegangan suplai. Posisi dan kecepatan dapat dikontrol dengan resolusi 1.024 *steps.* Menyediakan umpan balik untuk posisi sudut, kecepatan sudut, dan torsi beban.

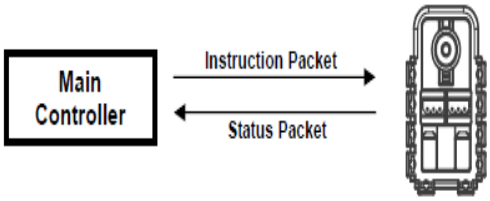
Aktuator *dynamixel* robot dapat memperingatkan pengguna bila parameter menyimpang dari rentang yang ditetapkan pengguna (suhu internal misalnya, torsi, tegangan, dan lain-lain) dan juga dapat menangani masalah secara otomatis (*off* torsi). Spesifikasi utama dari servo ini adalah:

Resolution	: 0,35°
Operating Angle	: 300°, *Endless Turn*
Voltage	: 7~10 V (*Recommended voltage*: 9,6 V)
Max. Current	: 900 mA
Operate Temperature	: -5 ~ +85^{o}C

Command Signal	: *Digital Packet*
Protocol Type	: *Half duplex Asynchronous Serial Communication*(8 bit, 1 *stop, No Parity*)
Link (Physical)	: *TTL Level Multi Drop (daisy chain type Connector)*
ID	: 254 ID (0~253)
Communication Speed	: 7343bps ~ 1 Mbps
Feedback	: *Position, Temperature, Load, Input Voltage, etc*.

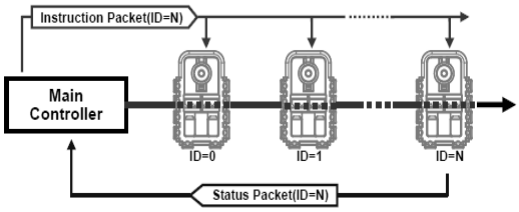
Untuk mengoperasikan aktuator *dynamixel*, pengendali utama harus didukung *TTL* tingkat *UART half duplex* (dianjurkan *CM*-5). Tiap *dynamixel* memiliki nama atau *ID* masing-masing. Apabila terjadi kesamaan nama pada lebih dari 1 *dynamixel*, maka akan terjadi *error*. Oleh karena itu, maka dianjurkan agar tidak memberi nama yang sama pada lebih dari 1 *dynamixel*.

Pengendali utama berkomunikasi dengan unit *dynamixel* dengan mengirim dan menerima paket data. Ada dua jenis paket: dikirim dari aktuator *dynamixel* ke kontroler utama disebut "*Instruction Packet*", dan dikirim dari kontroler utama ke aktuator *dynamixel* disebut "*StatusPacket*", seperti yang terlihat pada Gambar 6.



Gambar 6. Komunikasi paket

Untuk koneksisi stem pada contoh di bawah ini, jika pengendali utama mengirimkan sebuah paket instruksi dengan *set ID* untuk N, hanya unit *Dynamixel* dengan nilai *ID* tersebut yang akan kembali paket statusnya, dan melakukan instruksi yang diperlukan, seperti yang terlihat pada Gambar 7[6].



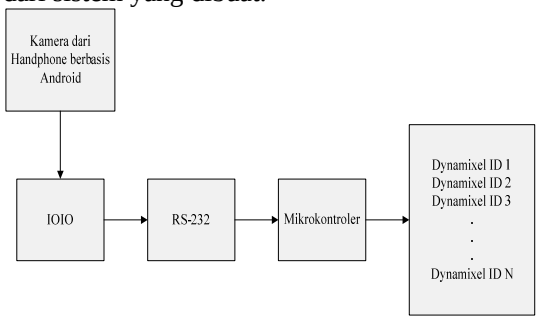
Gambar 7. Komunikasi untuk *ID N*

METODE PENELITIAN

Pada bagian ini, akan dibahas mengenai perancangan *hardware*, dan *software*. Perancangan *hardware* meliputi pembuatan *RS*-232 yang berfungsi sebagai komunikasi robot *humanoid* dengan *handphone* Android, lalu pembuatan mekanika gerak kepala robot. Untuk perancangan *software* meliputi pemrograman *Java* dari *handphone* Android, dan pemrograman robot menggunakan *roboplus*.

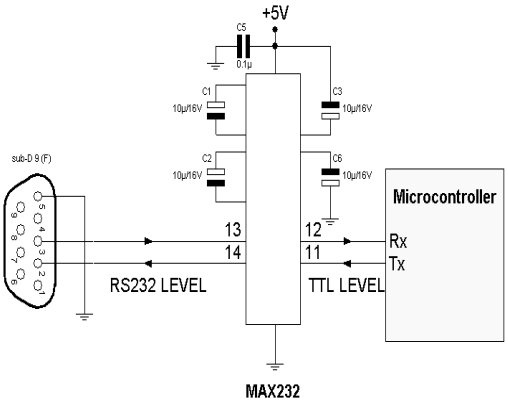
Perancangan *Hardware*

Pada Gambar 8 disajikan diagram blok dari sistem yang dibuat.



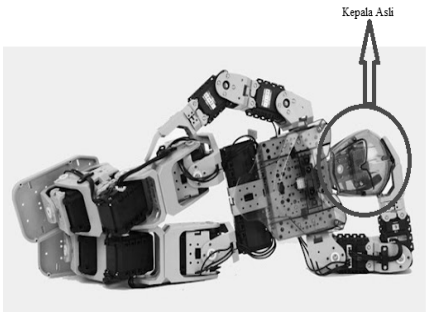
Gambar 8. Diagram blok sistem

Penjelasan bagian-bagian diagram blok sistem dari Gambar 8 sebagai berikut. Robot secara otomatis akan bergerak mencari objek setelah aplikasi yang di-*instal* pada Android diaktifkan. *Handphone* akan membaca setiap *pixel* yang tertangkap oleh kamera, dan akan mengirim perintah yang sudah ditentukan menuju *IOIO*. Setelah itu, *IOIO* akan mengirim sebuah *instruction packet* melalui jalur *UART* menuju *Max*-232. Pada rangkaian *RS*-232 ini sinyal yang dikirim diubah level tegangannya menjadi level tegangan *TTL*/*CMOS* agar dapat diterima oleh mikrokontrol *CM*-510, setelah itu sinyal akan dikirim menuju *CM*-510. Setelah *CM*-510 menerima *instruction packet* melalui *RS*-232, *CM*-510 meneruskan ke *dynamixel*-*dynamixel* yang terpasang. Perancangan *RS*-232 ini menggunakan *IC MAX*-232 untuk mengubah sinyal dari level tegangan *RS*-232 menjadi level *TTL*. Pengubahan level tegangan ini bertujuan agar perintah yang dikeluarkan dari Android dapat diterima dengan baik oleh mikrokontroler *CM*-510. Pada Gambar 9 disajikan rangkaian *RS*-232 yang digunakan.

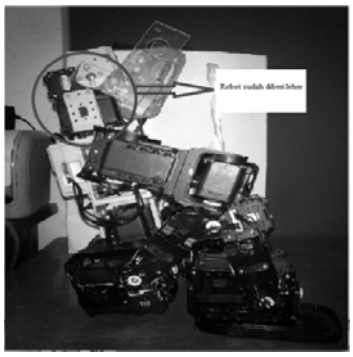


Gambar 9. Skematik rangkaian *RS*-232

Perancangan ini diperlukan untuk memperluas jarak pandang robot dalam mencari objeknya. Penambahan bagian pada robot ini menggunakan 2 *dynamixel AX*. Sebelum menggabungkan dua *dynamixel* tersebut, terlebih dahulu dilakukan pemberian nama *ID* pada keduanya. Dalam hal ini, *dynamixel* diberi nama *ID* 19, dan 20, sehingga tidak terjadi kondisi nama yang sama. Gambar 10 adalah gambar robot sebelum (a) dan sesudah penambahan bagian leher pada robot (b).

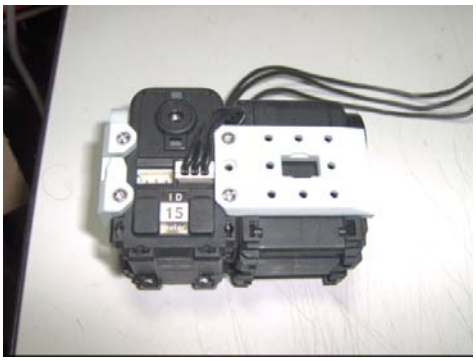


Gambar 10. (a) Kepala Asli



Gambar 10. (b) Kepala modifikasi

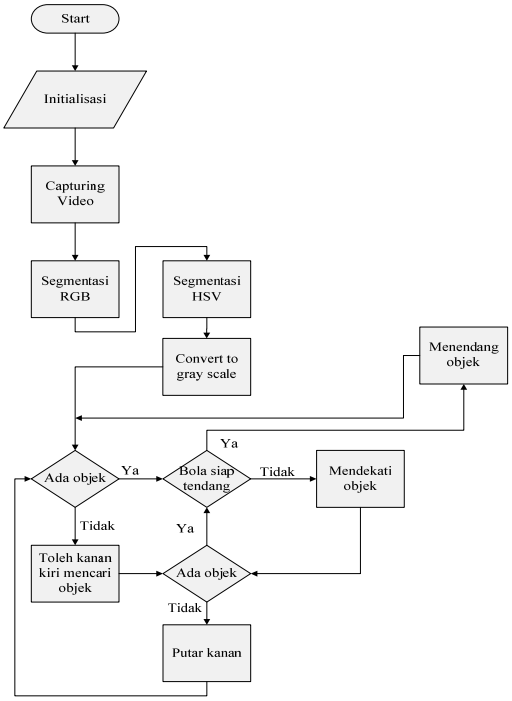
Perancangan leher dari robot menggunakan dua *dynamixel* yang disatukan dengan menggunakan *engineering plastic* yang sudah tersedia. Penggabungan dua *dynamixel* dapat dilihat pada Gambar 11.



Gambar 11. Dynamixel untuk leher robot

Perancangan *Software*

Bagian perancangan *software* meliputi penambahan program untuk menambah gerak dari leher robot, dan pembuatan aplikasi pada *handphone* Android. Penambahan program gerak pada robot agar robot dapat menoleh ke kanan, kiri, bawah, dan atas menggunakan program *roboplus*. Untuk pembuatan aplikasi pada Android menggunakan program *eclipse*. Pada Gambar 12 disajikan diagram alir utama program yang digunakan oleh penulis pada penelitian ini.



Gambar 12. Diagram alir utama program

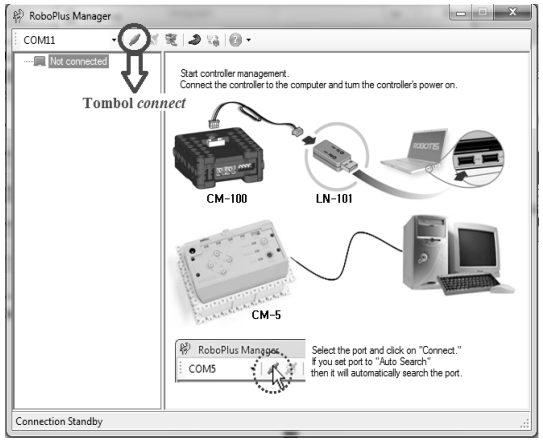
Penjelasan diagram alir utama, pada Gambar 12 diuraikan di bawah ini**:**

Mula-mula saat aplikasi pada Android dijalankan, maka aplikasi akan mulai menginisialisasi dan melakukan *capturing* video (robot harus dalam keadaan *stanby)*. Hasil dari *capturing* akan disegmentasi warna *RGB* dan *HSV*, kemudian di-*convert* ke *gray scale*. Semua benda dengan kondisi *Hue* antara 15-25, *Sat* antara 100-250, dan *Val* antara 20-200 (benda berwarna *orange*) akan diubah menjadi warna putih, sedangkan yang lainnya akan diubah menjadi warna hitam. Jika robot tidak melihat adanya benda yang menjadi targetnya, maka robot akan melakukan gerak toleh kanan kiri (robot mencari targetnya menggunakan cara menggerakkan kepalanya dengan urutan bawah tengah, bawah kiri, bawah kiri2, bawah kiri3, atas kiri3, atas kiri2, atas kiri, atas tengah, atas kanan, atas kanan2, atas kanan3, bawah kanan3, bawah kanan2, dan bawah kanan) untuk mencari targetnya. Apabila masih belum menemukan targetnya, maka robot akan putar kanan (robot dapat diprogram putar kiri, akan tetapi dalam kasus ini robot diprogram putar kanan karena robot lebih cepat berbalik arah dengan melakukan putar kanan) dan kemudian mencari kembali. Jika target sudah ditemukan, maka robot akan mendeteksi posisi dari target. Robot berjalan mendekati target hingga target berada pada jangkauan tendang. Saat target sudah berada pada jangkauan tendang, maka robot akan menendang targetnya. Sesudah menendang targetnya, robot akan mencari targetnya kembali, dan melakukan proses seperti sebelumnya. Kerja robot ini dilakukan secara berulang-ulang dan tidak pernah berhenti.

Penambahan Program Gerak Robot

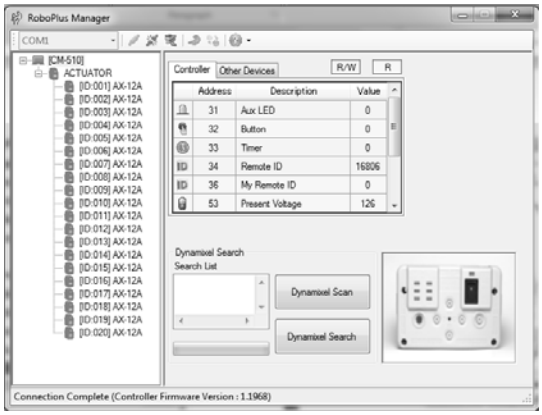
Untuk menambahkan program gerak dari robot digunakan program *roboplus*. Akan tetapi perlu dilakukan pemberian nama *ID* terlebih dahulu pada *dynamixel* yang akan ditambahkan. Langkah-langkah untuk pemberian nama *ID* pada *dynamixel* dijelaskan pada Gambar 13 dan 14.

Gambar 13. Tampilan *roboplus*

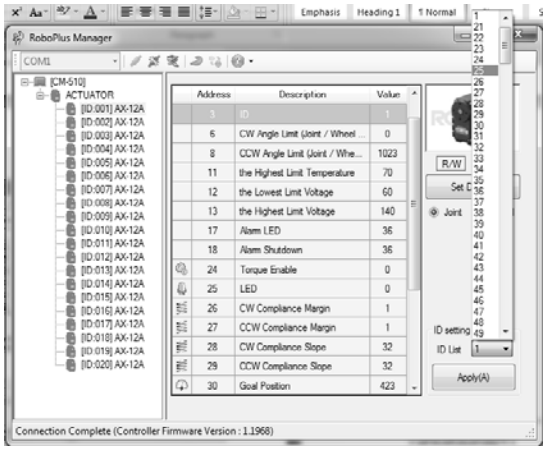


Gambar 14. Tampilan *roboplus manager*

Pertama-tama kondisi robot harus dalam keadaan *ON*, dan dihubungkan dengan komputer yang akan digunakan untuk memprogram. Kemudian jalankan program *roboplus*, lalu pilih *roboplus manager* (akan muncul tampilan seperti pada Gambar 13 kemudian Gambar 14). Sebelum di-*connect*, terlebih dahulu sesuaikan *com* yang terhubung dengan kabel serial yang digunakan. Untuk memeriksa *com* tersebut bisa dilihat pada *device manager* pada komputer. Setelah *connect*, maka akan muncul tampilan seperti Gambar 15. Setelah itu pilih *dynamixel* yang akan diberi nama *ID*-nya, setelah itu pilih *apply* sesudah selesai memberi nama *ID* (dapat dilihat pada Gambar 16). Dengan demikian pemberian nama *ID* telah selesai.



Gambar 15. Robot *connect* dengan *roboplus*

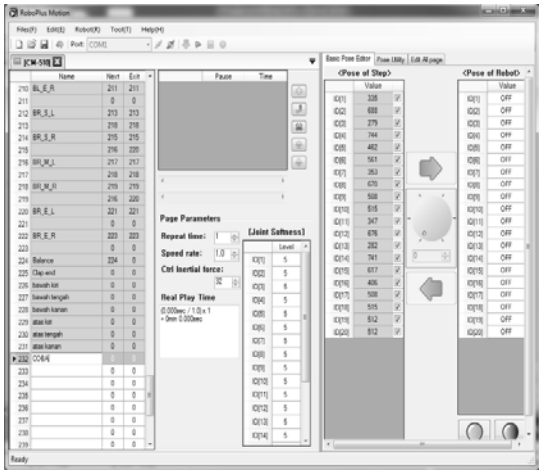


Gambar 16. Mengganti *ID dynamixel*

Setelah dipastikan semua *dynamixel* memiliki nama *ID*-nya masing-masing, buka kembali program *roboplus* lalu pilih *roboplus motion* seperti yang ditampilkan pada Gambar 17. Sebelum di-*connect*, kondisi robot harus *ON*, dan sudah dihubungkan ke *PC*, lalu di-*connect*.

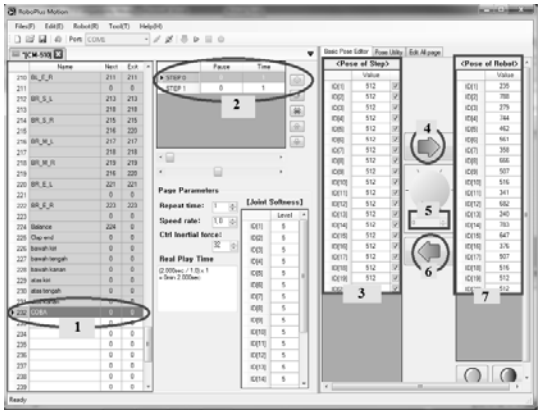
Gambar 17. Tampilan *roboplus*

Setelah robot dan *PC* telah *connect*, maka akan muncul tampilan seperti Gambar 18.



Gambar 18. Tampilan *roboplus motion*

Gambar 19 menunjukkan tampilan untuk membuat gerakan baru pada robot *humanoid*.



Gambar 19. Membuat gerakan baru

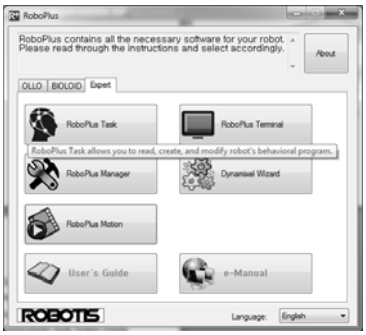
Penjelasan bagian-bagian dari Gambar 19 sebagai berikut:

1. Memberi nama untuk gerakan baru yang akan dibuat;
2. Menentukan jumlah *step* yang akan dibuat dalam satu gerakan;
3. Sudut tiap-tiap *dynamixel* yang ingin dipraktekkan;
4. Mengirim perintah kepada robot untuk melakukan *step* yang telah ditulis pada nomor 3;
5. Untuk mengganti sudut yang diinginkan pada tiap-tiap *dynamixel*,
6. Untuk men-*download* gerakan yang diinginkan pada robot;
7. Menunjukkan posisi sudut dari tiap-tiap *dynamixel* saat itu juga.

Untuk menciptakan gerakan baru pada robot, pertama-tama buat nama untuk gerakan baru tersebut. Langkah kedua adalah meng-*add step* sesuai dengan kebutuhan. Langkah ketiga,

mengisikan besarnya sudut pada tiap *dynamixel* (pengisian dilakukan pada tempat yang ditunjukkan dengan nomor 5) agar melakukan gerakan sesuai keinginan. Untuk dapat melihat gerakan robot sesuai keinginan atau belum, maka klik *button* yang ditunjukkan dengan nomor 4. Jika gerak robot belum sesuai dengan keinginan, maka atur ulang sudut dari tiap *dynamixel*. Namun apabila gerakan sesuai dengan yang diharapkan, maka klik *save* untuk menyimpan gerakan baru tersebut.

Setelah gerakan baru selesai dibuat, maka selanjutnya membuat program untuk menjalankannya setelah tombol dari *remote control* ditekan. Langkah pertama yaitu membuka *roboplus* lalu memilih *roboplus task* seperti yang ditunjukkan pada Gambar 20, dan Gambar 21. Sebelum memulai program, memilih terlebih dahulu tipe kontroler yang digunakan serta *port* yang tersambung dengan kabel serial dari robot. Setelah itu memulai membuat program. Jika program telah selesai dibuat, maka klik pilihan *download* program.



Gambar 20. *Roboplus task*

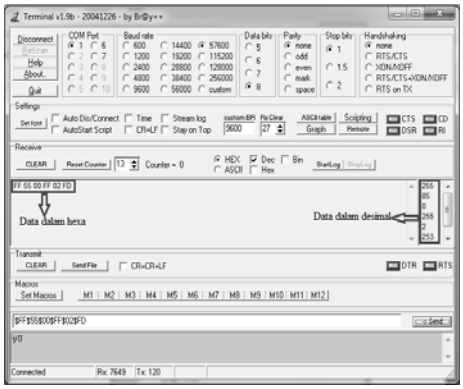
Gambar 21. Memprogram *roboplus task*

HASIL PENELITIAN DAN PEMBAHASAN

Pengukuran dilakukan untuk mengetahui kinerja dari alat yang telah didesain dan dibuat. Pengukuran dan pengujian yang dilakukan antara lain meliputi:

- **Pengujian *RS*-232**

Pengujian ini bertujuan untuk mengetahui pengiriman data oleh *RS*-232 sudah sesuai dengan yang diinginkan atau tidak. Pengujian ini dapat dilihat pada Gambar 22. Pada gambar tersebut, pengujian dilakukan dengan cara mengirim *intruction packet* (data serial) berupa FF55$00$FF02FD. Setelah data dikirim, data yang diterima pun sama dengan yang dikirim, yaitu FF 55 01 FE 00 FF (*hexa*), dan 255 85 0 255 2 253 (*decimal*).



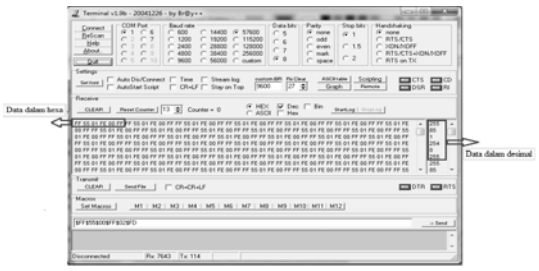
Gambar 22. Hasil pengujian *RS*-232

Dari Gambar 22 dapat disimpulkan bahwa *RS* 232 bekerja dengan baik untuk mengirimkan data dari *IOIO* menuju mikrokontroler *CM*-510.

- **Pengujian modul *IOIO***

Pengujian ini sangat diperlukan untuk mengetahui apakah *Instruction packet* yang dikirim dari *IOIO* menuju mikrokontroler *CM*-510 sesuai dengan yang dituliskan pada program. Hasil pengujian dapat dilihat pada Gambar 23.

Pengambilan data pada gambar tersebut dilakukan selama 5 detik, tetapi data yang diterima tampak sangat banyak serta hanya ada satu jenis paket instruksi saja. Pada mengujian ini menandakan bahwa data yang dikirimkan hanya satu paket instruksi saja, akan tetapi dikirim berulang-ulang dengan sangat cepat.

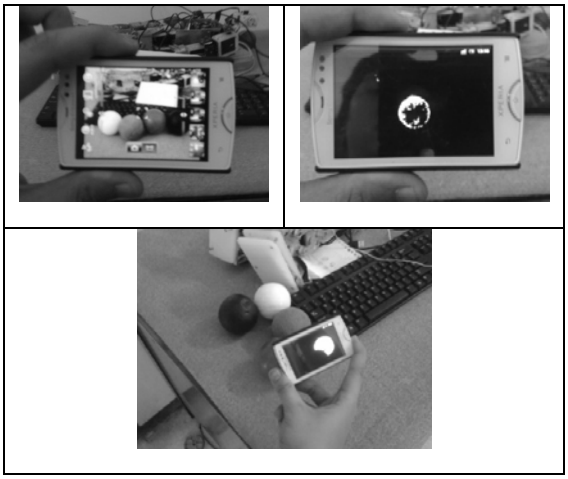


Gambar 23. Hasil pengujian pengiriman *instruction packet* dari *IOIO* menuju *CM*-510

- **Pengujian *Thresholding***

Agar dapat mengetahui kinerja *image processing* dari aplikasi yang telah dibuat pada Android tersebut, maka perlu dilakukan pengujian terlebih dahulu. Hasil dari pengujian tersebut dapat dilihat pada Gambar 24.

Pada Gambar 24 (a) menunjukkan gambar asli yang ditangkap oleh kamera dari *handphone* Android, sedangkan Gambar 24 (b) menunjukkan gambar yang telah melalui *image processing* oleh kamera dengan menggunakan aplikasi yang telah dibuat. Gambar 24 (c) menunjukkan gambar yang telah melalui *image processing* dengan posisi objek yang berbeda.



Gambar 24. (a) Gambar asli,
(b) Gambar asli yang telah di-*threshold*
(c) Hasil *threshold* dengan posisi objek sudah berubah

Dari gambar-gambar di atas, dapat dilihat bahwa gambar bola setelah dilakukan *threshold* tidak berbentuk bulat sempurna. Hal ini dikarenakan *range threshold* yang sempit, bertujuan agar warna yang tidak dikehendaki tidak ikut terdeteksi. Apabila ingin mendapatkan hasil berbentuk bulat penuh, maka *range* dari *threshold* harus dibuat lebih lebar, akan tetapi hal ini dapat mengakibatkan warna selain yang diinginkan dapat ikut terdeteksi.

- **Pengujian Kinerja Robot**

Pengujian ini bertujuan untuk mengetahui berapa lama waktu yang diperlukan robot untuk mencari, dan menendang targetnya, serta untuk mengetahui persentase keberhasilannya. Dalam pengujian ini akan dilakukan 4 variasi dari peletakan objek target, yaitu di depan robot, kanan robot, kiri robot, dan belakang robot. Jarak dari target juga akan dibuat menjadi dua macam, yaitu jarak 1 meter, dan 2 meter dari robot. Waktu untuk penyelesaian tugas diberi batas hingga 5 menit setelah robot *ON*, dan aplikasi Android dijalankan. Hasil pengujian kinerja robot disajikan pada Tabel 1, 2, 3, 4, 5, 6, 7, dan 8.

Tabel 1. Target di depan robot berjarak 1 meter

No.	Lama waktu (detik)	Keberhasilan
1	166	berhasil
2	154	berhasil
3	142	berhasil
4	174	berhasil
5	138	berhasil

Tabel 1 menunjukkan bahwa robot dapat berhasil 100% menyelesaikan tugas dengan waktu rata-rata 154,8 detik apabila target berada di depan robot, dan berjarak 1 meter dari robot.

Tabel 2. Target di kiri robot berjarak 1 meter

No.	Lama waktu (detik)	Keberhasilan
1	172	berhasil
2	179	berhasil
3	171	berhasil
4	187	berhasil
5	163	berhasil

Tabel 2 menunjukkan bahwa robot dapat berhasil 100% menyelesaikan tugas dengan waktu rata-rata 174,4 detik saat target berada di kiri robot, dan berjarak 1 meter dari robot.

Tabel 3. Target di kanan robot berjarak 1 meter

No.	Lama waktu (detik)	Keberhasilan
1	221	berhasil
2	211	berhasil
3	203	berhasil
4	199	berhasil
5	206	berhasil

Tabel 3 menunjukkan bahwa robot dapat berhasil 100% menyelesaikan tugas dengan waktu rata-rata 208 detik saat target berada di kanan robot, dan berjarak 1 meter dari robot.

Tabel 4. Target di belakang robot berjarak 1 meter

No.	Lama waktu (detik)	Keberhasilan
1	291	berhasil
2	287	berhasil
3	302	berhasil
4	281	berhasil
5	318	berhasil

Tabel 4 menunjukkan bahwa robot dapat berhasil 100% menyelesaikan tugas dengan waktu rata-rata 295,8 detik saat target berada di

belakang robot, dan berjarak 1 meter dari robot.

Tabel 5. Target di depan robot berjarak 2 meter

No.	Lama waktu (detik)	Keberhasilan
1	297	berhasil
2	316	berhasil
3	323	berhasil
4	309	berhasil
5	293	berhasil

Tabel 5 menunjukkan bahwa robot dapat berhasil 100% menyelesaikan tugas dengan waktu rata-rata 307,6 detik saat target berada di depan robot, dan berjarak 2 meter dari robot.

Tabel 6. Target di kiri robot berjarak 2 meter

No.	Lama waktu (detik)	Keberhasilan
1	308	berhasil
2	321	berhasil
3	294	berhasil
4	-	gagal
5	305	berhasil

Tabel 6 menunjukkan bahwa robot berhasil menyelesaikan tugas dengan prosentase keberhasilan sebesar 80% saat target berada pada kiri robot, dan berjarak 2 meter dari robot.

Tabel 7. Target di kanan robot berjarak 2 meter

No.	Lama waktu (detik)	Keberhasilan
1	293	berhasil
2	317	berhasil
3	291	berhasil
4	-	gagal
5	329	berhasil

Tabel 7 menunjukkan bahwa robot berhasil menyelesaikan tugasnya dengan prosentase keberhasilan sebesar 80% jika target berada pada kanan robot dan berjarak 2 meter dari robot.

Tabel 8. Target di belakang robot berjarak 2 meter

No.	Lama waktu (detik)	Keberhasilan
1	347	berhasil
2	-	gagal
3	329	berhasil
4	334	berhasil
5	-	gagal

Tabel 8 menunjukkan bahwa robot berhasil menyelesaikan tugasnya dengan prosentase keberhasilan sebesar 60% jika target berada di belakang robot, dan berjarak 2 meter dari robot.

Dari hasil pengujian di atas, dapat dihitung persentase keberhasilan robot menyelesaikan tugas. Dari 40 percobaan, berhasil 36 kali dan gagal 4 kali, maka persentase keberhasilannya adalah 90%.

KESIMPULAN

Dari hasil penelitian dan pembahasan yang melibatkan: perancangan, pembuatan, dan pengujian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut :

1. Pengolahan citra dapat dilakukan sebagai salah satu metode untuk robot pengidentifikasi benda;
2. Metode *thresholding* sangat mempengaruhi hasil *capturing video* untuk mencari sebuah benda yang menjadi targetnya;
3. *IOIO for* Android adalah sistem elekronika yang sangat membantu untuk menghubungkan *handphone* berbasis Android dengan peranti lain dari luar tanpa harus mengganti *software* ataupun menambahkan elektronika pada *handphone* tersebut;
4. Penambahan beban yang berlebih pada bagian kepala mengakibatkan robot *humanoid* menjadi kurang seimbang.

DAFTAR PUSTAKA

[1] Gonzalez, R.C., dan Wintz, P.*Digital Image Processing*, Edisi Kedua, Hlm. 55, 333, 595, Addison Wesley, New Jersey, 2002

[2] Sony Ericsson, *Xperia Mini: Si Mini yang Bertenaga*, http://www.idhandphone.com/ 2011/09/ sony-ericsson-xperia-mini-si-mini-yang.html, Diakses 2 maret 2012

[3] SparkFun, *IOIOfor Android Beginners Guide*.http://www.sparkfun.com/tutorials/280, diakses 3maret 2012.

[4] Ardiansyah, R., *In System Programing Menggunakan Koneksi Bluetooth*, Hlm. 11-13, Skripsi Jurusan Teknik Elektro, Fakultas Teknik Universitas Katolik Widya Mandala Surabaya, 2011

[5] Robotis Com, *Manual CM*-510, http://support.robotis.com/en/product/auxdevice/controller/cm510_manual.htm, Diakses 2 April 2012

[6] Robotis Com, *AX*-12/*AX*-12+/*AX*-12A, http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm, Diakses 2 April 2012

[7] Hermawan, S., *Mudah Membuat Aplikasi Android*, Penerbit Andi, Yogyakarta, 2011

[8] Mulyadi, A., *Pedoman Praktikum Pemrograman Java*, Surabaya, 2010

[9] Jain, A. K., *Fundamental of Digital Image Processing*, Prentice-Hall Inc., New Jersey, 1991